APPENDIX

```
/* final_x.c - final version of program */
#include <stdio.h>
#include <string.h>
#include <math.h>
#define NO_SEQ          500   /* max number of sequences */
#define NO_AA_SEG 50    /* max sequence length times 10 */

char garbage[40];
FILE *fp, *fo;

void ReadIn_Garbage(int y){
          int x;
          for (x=0; x<y; x++)
                fscanf(fp, "%s", garbage);
}

double factorial(float number){
        /* Good for values less than 170 */
        float gamma[101]={1.000, 99.433, 49.442, 32.785, 24.461,
19.470, 16.146,
                13.774, 11.997, 10.616, 9.514, 8.613, 7.863, 7.230,
6.689, 6.220,
                5.811, 5.451, 5.132, 4.847, 4.591, 4.360, 4.150, 3.960,
3.786,
                3.626, 3.478, 3.343, 3.217, 3.100, 2.992, 2.890, 2.796,
2.707,
                2.624, 2.546, 2.473, 2.404, 2.338, 2.277, 2.218, 2.163,
2.110,
                2.061, 2.013, 1.968, 1.925, 1.884, 1.845, 1.808, 1.772,
1.738,
                1.706, 1.675, 1.645, 1.616, 1.589, 1.562, 1.537, 1.513,
1.489,
                1.467, 1.445, 1.424, 1.404, 1.385, 1.366, 1.348, 1.331,
1.314,
                1.298, 1.282, 1.267, 1.253, 1.239, 1.225, 1.212, 1.200,
1.187,
                1.176, 1.164, 1.153, 1.142, 1.132, 1.122, 1.112, 1.103,
1.094,
                1.085, 1.077, 1.069, 1.061, 1.053, 1.046, 1.038, 1.031,
1.025,
                1.018, 1.012, 1.006, 1.000};
        double value;
        float t;
      int y;

        if (number > 20.0)
                value = exp(number*log(number)-number +
0.5*log(2*3.14159265*number));
        else if (number == 0.0)
            value = 1.0;
      else {
                value = 1.0;
                t = number;
```

```
                while (t > 1.0){
                        value = value * t;
                        t=t-1.0;
                }
                y=t*100;
                value = value * gamma[y] * t;
        }
        return (value);
}

double Calc_Value(int *root, int *fix, int *root_rand, int *fix_rand)
/* Function that calculates K1 */
{
        float mean[26]={0.072658, 0.000114, 0.024692, 0.050007,
0.061087,
                0.041774, 0.071589, 0.023392, 0.052691, 0.000000,
0.063923,
                0.089093, 0.023150, 0.042931, 0.000000, 0.052228,
0.039871,
                0.052012, 0.073087, 0.055606, 0.000000, 0.063321,
0.012720,
                0.000995, 0.032955, 0.000103};
        double K1=0.0, dEnergy, dEnergy_rand;
        int i;
        float th_root, th_fix;
        int total_fix=0, total_root=0;
        float th_root_rand, th_fix_rand;
        float total_fix_rand=0.0, total_root_rand=0.0;
      double K1_rand=0.0;

        for (i=0; i<26; i++){
                total_root+=root[i];
                total_fix+=fix[i];
                total_root_rand+=root_rand[i];
                total_fix_rand+=fix_rand[i];
      }

        for (i=0; i<26; i++){
            /* Calculates Regular Part */
                if (total_fix != 0)
                  th_fix = 274.0*fix[i]/total_fix;
            else
                  th_fix = 0.0;

            if (total_root != 0)
                        th_root = 274.0*root[i]/total_root;
                else
                        th_root = 0.0;
                dEnergy=0.0;

                if (mean[i] > 0.001){
                        dEnergy=dEnergy+(th_root-th_fix)*log(mean[i])+
                          (th_fix-th_root)*log(1-mean[i]);

                  if (th_fix > 0.01)
                        if (th_fix > 170.0)
```

```
                                    dEnergy = dEnergy +
(th_fix*log(th_fix)-th_fix);
                              else
                                    dEnergy = dEnergy +
log(factorial(th_fix));

                        if (th_root > 0.01)
                              if (th_root >170.0)
                                    dEnergy = dEnergy -
(th_root*log(th_root)-th_root);
                              else
                                    dEnergy = dEnergy -
log(factorial(th_root));

                        if (274.0 - th_fix > 0.01)
                        if ((274.0 - th_fix) > 170.0)
                                    dEnergy = dEnergy + ((274.0-th_fix)*
                                  log(274.0-th_fix)-(274.0-th_fix));
                              else
                                    dEnergy = dEnergy +
log(factorial(274.0-th_fix));

                        if (274.0 - th_root > 0.01)
                              if ((274.0 - th_root) >170.0)
                                    dEnergy = dEnergy - ((274.0-th_root)*
                                  log(274.0-th_root)-(274.0-th_root));
                              else
                                    dEnergy = dEnergy -
log(factorial(274.0-th_root));

            }

            /* Calculates Random Part */
                  if (total_fix_rand != 0)
                  th_fix_rand = 274.0*fix_rand[i]/total_fix_rand;
            else
                  th_fix_rand = 0.0;

            if (total_root_rand != 0)
                              th_root_rand =
274.0*root_rand[i]/total_root_rand;
                  else
                              th_root_rand = 0.0;
                  dEnergy_rand=0.0;
                  if (mean[i] > 0.001){
                              dEnergy_rand=dEnergy_rand+(th_root_rand-
th_fix_rand)*log(mean[i])+
                                    (th_fix_rand-th_root_rand)*log(1-
mean[i]);

                              if (th_fix_rand > 0.01)
                              if (th_fix_rand > 170.0)
                                    dEnergy_rand = dEnergy_rand +
(th_fix_rand*log(th_fix_rand)-th_fix_rand);
                              else
                                    dEnergy_rand = dEnergy_rand +
log(factorial(th_fix_rand));
```

```
                        if (th_root_rand > 0.01)
                        if (th_root_rand >170.0)
                                dEnergy_rand = dEnergy_rand -
(th_root_rand*log(th_root_rand)-th_root_rand);
                        else
                                dEnergy_rand = dEnergy_rand -
log(factorial(th_root_rand));

                        if (274.0 - th_fix_rand > 0.01)
                        if ((274.0 - th_fix_rand) > 170.0)
                                dEnergy_rand = dEnergy_rand + ((274.0-
th_fix_rand)*
                                        log(274.0-th_fix_rand)-(274.0-
th_fix_rand));
                        else
                                dEnergy_rand = dEnergy_rand +
log(factorial(274.0-th_fix_rand));

                        if (274.0 - th_root_rand > 0.01)
                        if ((274.0 - th_root_rand) >170.0)
                                dEnergy_rand = dEnergy_rand - ((274.0-
th_root_rand)*
                                        log(274.0-th_root_rand)-(274.0-
th_root_rand));
                        else
                                dEnergy_rand = dEnergy_rand -
log(factorial(274.0-th_root_rand));

                }
            K1_rand+=dEnergy_rand*dEnergy_rand;
            dEnergy=dEnergy-dEnergy_rand;
            K1+=dEnergy*dEnergy;
      }
      K1=sqrt(K1);
        return (K1);
}



main()
{
      FILE *fs, *ft, *fh;
            int atom_no[3000], aa_no[3000];
        float pos_x[3000], pos_y[3000], pos_z[3000], occup[3000];
      float B_fac[3000];
        char atom[3000][4], aa[3000][4], chain[3000][2];
      int t=0;
      double mean_val[NO_SEQ];
        int temp;
      double K1;
      int aacount[500][27];
      int aacount_fix[500][27];
      char seqname[500][22];
        char seq[NO_SEQ][NO_AA_SEG][10];
        char datain[12],dataout[12];
```

```
        int no_rows;
        int  aaname, aanum;
              int s,z,x,y=0;
          int pno=0, mlen, len, fix,no;
          float shit;
        int sel, seqflag[500];
        char tempc[10];
        int atom_pdb,aa_pdb;
        int aa_dist[26];
        int aa_dist_fix[26];
        char sain[20];
        char pdbin[20];
        int before, after;
        int numbers;

        printf("Enter structure/alignment file: ");
        scanf("%s", sain);
        fh=fopen(sain, "r");
        fscanf(fh, "%s", pdbin);

/*  Sets Up I/O Files  */
            printf("Enter input filename: ");
            scanf("%s", datain);
            fp=fopen(datain, "r");
            printf("Enter outfile: ");
            scanf("%s", dataout);
            fo=fopen(dataout, "w");

/*  Reads in Header of msf file */

            strcpy(garbage, "duh");
          while (strcmp(garbage, "MSF:"))
                ReadIn_Garbage(1);

            fscanf(fp, "%d", &len);

          while (strcmp(garbage, "Name:"))
                ReadIn_Garbage(1);

/*  Calculates mlen and no_rows from len  */
            mlen=len/10;
            if (mlen*10 != len)
                    mlen++;
          no_rows=mlen/5;
          if (no_rows*5 != mlen)
                no_rows++;
          printf("no_rows = %d ", no_rows);

/*  Reads in Sequence names  */
            no=0;
          while (strcmp(garbage, "//")){
                    fscanf(fp, "%s", seqname[no++]);
                strcpy(garbage,"duh");
                while (strcmp(garbage, "Name:") && strcmp(garbage, "//"))
                      ReadIn_Garbage(1);
          }
```

```
/*  Reads sequence into array  */
      fscanf(fp, "%s", garbage);
      if (strcmp(garbage, "1"))
            numbers=0;
      else
            numbers=1;

      for (z=0; z<no_rows; z++){
            if (numbers)
                  ReadIn_Garbage(2);
                for (y=0; y<no; y++){
                      for (x=0; x<5 && 5*z+x < mlen; x++)
                                  fscanf(fp, "%s", seq[y][(x)+5*(z)]);
                  ReadIn_Garbage(1);
            }
      }


/* Converts all lowercase to uppercase */

      for (z=0; z<len; z++)
            for (y=0; y<no; y++)
                  if (seq[y][0][z] >= 'a' && seq[y][0][z] <= 'z')
                        seq[y][0][z]=seq[y][0][z]+('A' - 'a');

/*  Selection for/against amino acids at a position  */
      for (x=0; x<2; x++){
            if (x)
                  printf("Selection against amino acids (x when
done)\n");
            else
                  printf("Selection for amino acids (x when done)\n");
            t=1;
            do {
                  printf("Enter amino acid: ");
                  scanf("%s", tempc);
                  aaname=tempc[0];
                  if (aaname >= 'a' && aaname <= 'z')
                        aaname = aaname + ('A' - 'a');
                  if (aaname == 'X' && !x && t)     /* Finish that
selection */
                        for (y=0; y<no; y++)
                                    seqflag[y]=1;
                  if (aaname != 'X'){
                              printf("Enter amino acid number: ");
                              scanf("%d", &aanum);
                        for (z=0; z<no; z++)
                              if (seq[z][0][aanum-1] == aaname)
                                    seqflag[z]=1-x;
                  }
                  fprintf(fo,"%d - AA=%c, AA#=%d\n", 1-x, aaname,
aanum);
                  t=0;
            } while (aaname != 'X');
      }
```

```
      fprintf(fo, "\n");


/*  Reads in pdb file  */

        fs=fopen(pdbin, "r");
        ft=fopen(strcat(dataout,".pdb"), "w");

        do{
                fscanf(fs, "%s", garbage);
        } while (strcmp(garbage, "ATOM"));

        x=0;
        do{
                fscanf(fs, "%d", &atom_no[x]);
                fscanf(fs, "%s", atom[x]);
                fscanf(fs, "%s", aa[x]);
                fscanf(fs, "%s", chain[x]);
                fscanf(fs, "%d", &aa_no[x]);
                fscanf(fs, "%f", &pos_x[x]);
                fscanf(fs, "%f", &pos_y[x]);
                fscanf(fs, "%f", &pos_z[x]);
                fscanf(fs, "%f", &occup[x]);
                fscanf(fs, "%f", &B_fac[x++]);
                fscanf(fs, "%s", garbage);
        } while (strcmp(garbage, "END"));
      atom_pdb=x;
      aa_pdb=aa_no[x-1]-aa_no[0]+1;


/*  Count amino acids/position  */
      for (x=0; x<len; x++)
            for (y=0; y<27; y++){
                  aacount[x][y]=0;
                  aacount_fix[x][y]=0;
            }
        for (x=0; x<mlen; x++)
                for (z=0; z<10; z++)
                for (y=0; y<no; y++){
                        if (seq[y][x][z] >= 'A' && seq[y][x][z] <= 'Z')
                                aacount[x*10+z][seq[y][x][z]-'A']++;
                        else if (seq[y][x][z] == '.')
                                aacount[x*10+z][26]++;
                        if (seqflag[y]){
                              if (seq[y][x][z] >= 'A' && seq[y][x][z]
<= 'Z')
                                    aacount_fix[x*10+z][seq[y][x][z]-
'A']++;
                        else if (seq[y][x][z] == '.')
                                    aacount_fix[x*10+z][26]++;
                    }
            }

      temp=0;
        s=0;
        for (x=0; x<26; x++){
```

```
                aa_dist[x]=0;
                aa_dist_fix[x]=0;
        }

        for (x=0; x<26; x++)
                for (y=0; y<len; y++){
                        aa_dist[x]+=aacount[y][x];
                        aa_dist_fix[x]+=aacount_fix[y][x];
                }

      for (y=0; y<aa_pdb; y++){
            if (s>before || y == 0){
                  fscanf(fh, "%d", &before);
                  if (before || y==0)
                        fscanf(fh, "%d", &after);
            }

            if (s == before)
                  s = after;

            K1=Calc_Value(aacount[s], aacount_fix[s], aa_dist,
aa_dist_fix)/100.0;
            mean_val[y]=K1;
            s++;
      }

      /*  Writes filenames to output file  */
      for (z=0; z<no_rows; z++){
                for (y=0; y<no; y++){
                          if (seqflag[y]){
                              fprintf(fo, "%-10s", seqname[y]);
                              for (x=0; x<5 && x+5*z!=mlen; x++)
                              fprintf(fo, "%.10s ", seq[y][(x)+(5*z)]);
                              fprintf(fo, "\n");
                              pno++;
                          }
                }
            fprintf(fo, "\n\n");
        }
      printf("no_rows = %d ", no_rows);
        pno=pno/no_rows;

      /*  Print AA Composition */
        fix=1;
        do{
            printf("\nAA Comp which position (0 to exit)? ");
                scanf("%d", &fix);
                if (fix != 0){
                        for (x=0; x<26; x++) {
                              shit=100*aacount_fix[fix-1][x]/pno;
                              printf("%c = %3d (%3.0f%%)  ", x +'A',
                              aacount_fix[fix-1][x], shit);
                        }
                        shit=100*aacount_fix[fix-1][26]/pno;
                        printf(". = %3d (%3.0f%%)\n", aacount_fix[fix-
1][26],shit);
                }
```

```
        } while (fix != 0);

      temp=0;
      s=0;

/*  Writes out new PDB file with K1  */
        fprintf(ft, "GRASP PDB FILE\n");
        fprintf(ft, "FORMAT NUMBER=3\n");
      for (y=0; y<atom_pdb; y++){
            if (temp != aa_no[y])
                  K1=mean_val[s++];
          fprintf(ft,"ATOM   %4d  %-4s%s %s %3d     %7.3f %7.3f \
%7.3f ",  atom_no[y], atom[y], aa[y],chain[y],aa_no[y], pos_x[y],
pos_y[y], pos_z[y]);
            fprintf(ft," %5f\n", K1);
            temp=aa_no[y];
        }
      fprintf(ft, "END");

}
```

```
/*  Creates an SA file that containes information about how to align
the msf to the structure defined.  This requires there be no deleted
amino acids in the alignment or pdb. This assumes there will be x and
x+1 for each aa in the pdb. Starting position is based on the position
in the alignment the corresponds to the first position in the pdb file
- */

#include <stdio.h>

char garbage[40];
FILE *fp, *fo;

void ReadIn_Garbage(int y){
          int x;
          for (x=0; x<y; x++)
                fscanf(fp, "%s", garbage);
}


main()
{
FILE *fs, *ft;
      int atom_no[3000], aa_no[3000];
        float pos_x[3000], pos_y[3000], pos_z[3000], occup[3000],
B_fac[3000];
        char atom[3000][4], aa[3000][4], chain[3000][2];
      int t;
      char seqname[500][22];
        char seq[500][50][10];
        char datain[12],dataout[12];
        char name[20]="Name:";
            int s,z,x,y=0;
        int mlen, len,no;
      int i,j;
      int atom_pdb,aa_pdb;
      char searchname[22];
      int startaa;
      int nameno;
      int temp;
      int numbers;
      int no_rows;

/*  Sets Up I/O Files  */
        printf("Enter input alignment filename: ");
        scanf("%s", datain);
        fp=fopen(datain, "r");
      printf("Enter input PDB file: ");
      scanf("%s", datain);
      fs=fopen(datain, "r");
      printf("Enter corresponding name of structure to alignment: ");
      scanf("%s", searchname);
      printf("Enter starting position on alignment: ");
      scanf("%d", &startaa);
      printf("Enter outfile: ");
        scanf("%s", dataout);
        fo=fopen(dataout, "w");
```

```
/*  Reads in Header  */

        strcpy(garbage, "duh");
      while (strcmp(garbage, "MSF:"))
            ReadIn_Garbage(1);

        fscanf(fp, "%d", &len);

      while (strcmp(garbage, "Name:"))
            ReadIn_Garbage(1);


/*  Calculates mlen from len  */
        mlen=len/10;
        if (mlen*10 != len)
                mlen++;
      no_rows=mlen/5;
      if (no_rows*5 != no_rows)
            no_rows++;

/*  Reads in Sequence names  */
        no=0;
      while (strcmp(garbage, "//")){
                fscanf(fp, "%s", seqname[no++]);
            strcpy(garbage,"duh");
            while (strcmp(garbage, "Name:") && strcmp(garbage, "//"))
                  ReadIn_Garbage(1);
      }


/*  Reads sequence into array  */
      fscanf(fp, "%s", garbage);
      if (strcmp(garbage, "1"))
            numbers=0;
      else
            numbers=1;

      for (z=0; z<no_rows; z++){
            if (numbers)
                  ReadIn_Garbage(2);
                for (y=0; y<no; y++){
                      for (x=0; x<5 && 5*z+x < mlen; x++){
                                  fscanf(fp, "%s", seq[y][(x)+5*(z)]);
                  }
                  ReadIn_Garbage(1);
            }
      }


/* Converts all lowercase to uppercase */

      for (z=0; z<len; z++)
            for (y=0; y<no; y++)
                  if (seq[y][0][z] >= 'a' && seq[y][0][z] <= 'z')
                        seq[y][0][z]=seq[y][0][z]+('A' - 'a');
                  else if (seq[y][0][z] < 'A' || seq[y][0][z] > 'Z')
                        seq[y][0][z]='.';
```

```
/*  Reads in pdb  */


        do{
                fscanf(fs, "%s", garbage);
        } while (strcmp(garbage, "ATOM"));

      temp=0;
        x=0;
        do{
                fscanf(fs, "%d", &atom_no[x]);
                fscanf(fs, "%s", atom[x]);
                fscanf(fs, "%s", aa[x]);
                fscanf(fs, "%s", chain[x]);
                fscanf(fs, "%d", &aa_no[x]);
                fscanf(fs, "%f", &pos_x[x]);
                fscanf(fs, "%f", &pos_y[x]);
                fscanf(fs, "%f", &pos_z[x]);
                fscanf(fs, "%f", &occup[x]);
                fscanf(fs, "%f", &B_fac[x]);
            do{
                  fscanf(fs, "%s", garbage);
            }while (strcmp(garbage,"ATOM") && strcmp(garbage, "END"));
            if (temp != aa_no[x])
                  temp=aa_no[x++];
        } while (strcmp(garbage, "END"));
      atom_pdb=x;

/*  Finds corresponding index for alignment name */
      nameno=-1;
      for (i=0; i<no; i++)
            if (!strcmp(seqname[i],searchname))
                  nameno=i;
      if (nameno == -1)
            printf("Sequence not found!!!!!!\n");

      startaa--;

      fprintf(fo, "%s\n", datain);
      if (startaa !=0)
            fprintf(fo, "0  %d\n", startaa);
      x=0;
      for (i=startaa; i<len || x<atom_pdb; i++)
            if (seq[nameno][0][i] != '.' && aa_no[x+1] == aa_no[x]+1)
                  x++;
            else if (aa_no[x+1] != aa_no[x]+1){
                  s=i;
                  do {
                        x++;
                        i++;
                  } while (aa_no[x+1] != aa_no[x]+1);
                  t=i;
                  if (x<atom_pdb)
                        fprintf(fo, "%d  %d\n", s,t);
            }
            else if (seq[nameno][0][i] == '.'){
```

```
                    s=i;
                    do
                          i++;
                    while (seq[nameno][0][i] == '.');
                    t=i;
                    if (x<atom_pdb && t != len)
                          fprintf(fo, "%d  %d\n", s,t);
              }
        fprintf(fo, "000\n");

/*  Closes files  */
          fclose(fo);
          fclose(fp);
        fclose(fs);
        fclose(ft);

}
```